Unsupervised Aspect Discovery from Online Consumer Reviews

by

Kaheer Suleman

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

Abstract

The success of on-line review websites has led to an overwhelming number of on-line consumer reviews. These reviews have become an important tool for consumers when making a decision to purchase a product. This growth has led to the need for applications that enable this information to be presented in a way that is meaningful. These applications often rely on domain specific semantic lexicons which are both expensive and time consuming to make.

The following thesis proposes an unsupervised approach for product aspect discovery in on-line consumer reviews. We apply a two step hierarchical clustering process in which we first cluster based on the semantic similarity of the contexts of terms and then on the similarity of the hypernyms of the cluster members. The method also includes a process for assigning class labels to each of the clusters. Finally an experiment showing how the proposed methods can be used to measure aspect based sentiment is performed.

The methods proposed in this thesis are evaluated on a set of 157,865 reviews from a major commercial website and found that the two-step clustering process increases cluster F-scores over a single round of clustering. Finally, the proposed methods are compared to a state of the art topic modelling approach by Titov and McDonald (2008).

Acknowledgements

I would like to thank my supervisor, Dr. Olga Vechtomova, for the support and guidance provided throughout the process of writing this thesis.

I would also like to thank Dr. Charlie Clarke and Dr. Gordon Cormack for accepting to read my thesis.

Finally I would like to express my gratitude to Sharon Choy and Jack Thomas for helping with the annotation of the evaluation data sets.

Table of Contents

List of Tables viii

List of Figures ix

1 Introduction 1

2 Background and Related Work 4

2.1 Opinion and Aspect Extraction 4

2.2 Supervised Methods 5

2.2.1 Sequential Classification 5

2.2.1.1 Markov Models 5

2.2.1.2 Conditional Random Fields 7

2.2.2 Other Supervised Methods 8

2.2.3 Semi Supervised and Unsupervised Methods 9

2.2.3.1 Frequency based Methods 9

2.2.3.2 Pattern Based Methods 9

2.2.3.2.1 Dependency Parsing 11

2.2.3.3 Clustering Based Methods 11

2.2.3.4 Model Based Approaches 13
2.2.3.4.1 Classification Based Approaches 13
2.2.3.4.2 Topic Modelling Approaches 13
2.2.3.4.3 Multi grain Topic Modelling 14
2.3 Semantic Lexicon Construction 16
2.3.1 Term Similarity 16
2.3.2 Clustering Methods 18
2.3.2.1 Automatic Labeling of Clusterings 20
2.3.3 Pattern Based Methods 20
2.3.4 Bootstrapping Approaches 22

3 Methodology **25**
3.1 Candidate List Generation 26
3.2 Clustering 27
3.2.1 Lin's Similarity 27
3.2.2 Hierarchical Clustering 29
3.3 Merging Clusters based on Hypernyms 31
3.3.1 Motivation 31
3.3.1.0.1 Algorithm 31
3.3.2 Generation of Extraction Patterns 31
3.3.3 Extraction of Hypernym Candidates 32
3.3.4 Merging of Clusters based on Hypernyms 32
3.4 Aspect Oriented Sentiment 34
3.4.0.0.2 Algorithm 35

3.4.1 Lexicon Creation 35
3.4.2 Modifier Extraction 36
3.4.3 Aspect Polarity Scoring 36
3.4.4 SentiWordNet 36
3.4.5 Multiword Units 37

4 Results and Discussion **40**

4.1 Experimental Setup 40
4.1.1 Preprocessing 41
4.2 Evaluation 41
4.2.1 Clustering Without Hypernyms 41
4.2.2 Clustering With Hypernyms 43
4.2.3 Cluster Labeling 44
4.3 Aspect Based Sentiment 46
4.3.1 Kulback Leibler Divergence 46
4.4 Comparison to Multigrain Topic Modeling 48

5 Conclusion and Future Work **51**

References **54**

List of Tables

2.1 Example Heuristics used by AutoSlog (Riloff,1996) 24

3.1 Example Dependency Triples with Part of Speech 26
3.2 Modifier Dependency Relations 27
3.3 Example Similarity Matrix 30
3.4 Example of adapting Hearst's rules for term "pizza" 32
3.5 Clusters and their Top 5 Ranked Hypernyms 33
3.6 List of seed terms for food related terms 35

4.1 Cluster F_1 scores including non-cluster terms 43
4.2 Cluster F_1 scores excluding non-cluster terms 43
4.3 Maximum F_1 score for various stopping thresholds (including non-cluster terms) 44
4.4 Maximum F_1 score for various stopping thresholds (excluding non-cluster terms) 44
4.5 Precision and Recall for Automatically Generated Food Lexicon 45
4.6 Comparison of Precision for various Polarity Score Calculations 47
4.7 Comparison of Recall for various Polarity Score Calculations 48
4.8 Cluster F_1 scores for MGLDA vs Hierarchical Clustering 49

List of Figures

2.1 Graph structures of various conditional random fields 8

3.1 Diagram of Hierarchical Clustering Process 30

4.1 Pseudocode for evaluating aspect oriented sentiment 47

Chapter 1

Introduction

The success of on-line review services such as Zagat, TripAdvisor and Yelp has led to the presence of large numbers of on-line consumer reviews. These reviews have become an important factor to consumers for deciding to make a purchase. A 2013 study by the Business Development Bank of Canada suggested that 40% of individuals consult online reviews when deciding to make a purchase and that 70% of those individuals trust the contents in the reviews. The study also found that 7 out of 10 individuals had decided not to make a purchase based on a negative review. Evidence such as this makes it clear that it is important that providers of services, similar to those mentioned above, present the information in reviews in a way that consumers are easily able to understand.

One such method that has been successful is to present a summary based on the rateable aspects associated with the review domain. Aspects are defined as *the properties of an object that can be commented on by a reviewer* (Titov and McDonald, 2008; Synder and Barzilay, 2007). An example of this can be seen on TripAdvisor [1]. Here, each hotel reviewer is asked to give a rating for each of following aspects:

1. Service
2. Value

[1]http://www.tripadvisor.com

3. Sleep Quality

4. Cleanliness

5. Location

6. Rooms

7. Breakfast

Creating this list of aspects manually can be a time consuming and expensive task that would require domain expertise. While this may be possible for a service provider that only focusses on a small number of domains this would be infeasible for a highly varied list of products such as that of Amazon[2]. It is, therefore, important to develop applications which can process and analyze reivews and present the information in an automatic fashion. These applications often rely on the availability of domain specific semantic lexicons. Domain independent lexicons such as WordNet (Miller, 1990), although widely available, often lack domain specific terms such as proper nouns and jargon. While it is possible to generate domain specific lexicons by hand, this can be both a time consuming and expensive exercise. Furthermore, it is often a requirement that the classes are known ahead of time which may not be possible in some applications. For example, while one could come up with a number of common aspects for hotels, it would be difficult to determine a list of aspects associated with electric shavers. In order to be able to handle the variety of products in a review repository such as that of Amazon, any such method must have the following properties:

1. It must be able to identify a large variety of terms belonging to different aspects;

2. It must be able to provide a label for each aspect in order to assist application developers;

3. It must be scalable to a large number of products.

[2]http://www.amazon.com

In this thesis, we propose an unsupervised method for discovering semantic classes in on-line consumer reviews which we believe satisfies the criteria presented at the end of the previous paragraph. Our methods are based on the underlying assumption that terms with similar meanings appear in similar context (Lin,1998). We propose a 2-phase clustering process to identify nouns relating to rateable aspects found in consumer reviews. In the first phase of our method, we apply agglomerative hierarchical clustering to group similar terms together based on the contexts they appear in the reviews. In the second phase we transform each of the clusters into a higher level representation by building clusters based on the hypernyms of the cluster members. We, then, further cluster our existing clusters based on the higher level representations. One of the criticisms of unsupervised methods such as clustering is that they do not provide a method for assigning class labels. In this thesis, we propose a method for automatically, assigning class labels based on the hypernyms of the cluster members. Finally, we propose an approach for how our methods could be used to assist an application for identifying aspect based sentiment.

We evaluated our system on a set of 157,865 restaurant reviews. We evaluate each step in our method in order to isolate errors that could potentially be transferred from step to step. We also evaluate our method as part of an application of aspect oriented sentiment using a variety for calculating polarity of a term based on SentiWordnet (Esuli and Sebastiani, 2006). Finally, we compare our approach for aspect discovery to a state of the art topic modeling system proposed by Titov and Macdonald (2008).

The rest of this thesis is structured as follows. In Chapter 2 we review the previous work in the areas of semantic class discovery and product aspect extraction. In Chapter 3 we describe, in detail, our approach for aspect discovery and how our methods can be applied for the purpose of aspect oriented sentiment analysis. In Chapter 4 we describe our evaluation methodology and discuss the results of our experiment and in Chapter 5 we conclude and discuss possible areas for future work.

Chapter 2

Background and Related Work

The work presented in this thesis can fall under two general categories: product aspect extraction and semantic class discovery. The goal of product aspect extraction is to identify terms and phrases related to distinct features of products. Examples of aspects include phrases such as “battery life” for mobile phones and “ambience” for restaurants. Methods for semantic class discovery attempt to uncover groups of terms that relate to distinct semantic concepts such as automobiles, animals and food. The following section will review a section of the previous approaches in both of these areas.

2.1 Opinion and Aspect Extraction

The explosive growth in online consumer reviews has given rise to an extensive number of works in the area of opinion and aspect extraction. These works can be broken down into 3 main categories: supervised, semi supervised and unsupervised. Supervised methods rely on annotated corpora for training of statistical machine learning methods (Blair-Goldensohn et al., 2008; Jin et al., 2009; Qi, 2009; Li et al., 2010; Jakob and Gurevych, 2010; Yu et al., 2011). The training corpora used for these methods can be both time-consuming and costly to create; therefore, there has been an emphasis on semi supervised and unsupervised techniques. The following section will provide an overview of the various techniques.

2.2 Supervised Methods

2.2.1 Sequential Classification

Supervised approaches for the task of extracting products, features and opinions often view the problem as a sequential tagging problem similar to named entity recognition (Borthwick, 1999). Each term, in a review, is tagged according to its role. The terms are usually classified into whether or not they belong to a phrase representing a product or opinion as well as their role in the phrase (Qi, 2009; Li et al., 2010; Jin et al., 2009). Various sequential classification methods such as hidden markov models (HMM) (Jin et al., 2009) and conditional random fields (CRF) (Qi, 2009; Lin et al., 2010; Jakob and Gurevych, 2010) have been applied to this problem.

2.2.1.1 Markov Models

HMMs have been applied to a large number of problems in text and speech processing such a speech recognition (Rabiner, 1990), part of speech tagging (Kupiec, 1992; Charniak et. al. 1993), information extraction (Bikel et. al., 1997) and word segmentation (Lafferty, 2001). They consist of two variables: the "hidden" or state variables (S) and the evidence variables (E). The joint probability is written as follows:

$$P(S_{0:N}, E_{1:N}) = P(S_o) \prod_{n} P(Sn \| S_{n-1:0}) P(E_n \| E_{n-1:1}, S_{n:1}) \tag{2.1}$$

In Equation 2.1, the probabilities for moving from one state to another are defined by the transition probabilities. In order to simplify the model, the following 2 assumptions are applied:

1. The current state is independent of all previous states given the previous n states (markov assumption).
2. The evidence is independent of all other previous states given the current state.

After applying the above assumptions ($n = 1$) to both the evidence and the state variables, Equation 2.1 can be rewritten as:

$$P(S_{0:N}, E_{1:N}) = P(S_o) \prod_n P(S_n \| S_{n-1}) P(E_n \| S_n) \tag{2.2}$$

Traditionally, HMMs only support simple features such as words,however, lexical information is often important for aspect extraction. Jin et. al., (2009) propose that lexicalized HMMs (Lee et al., 2000; Fu and Luke, 2005) can be used in order to address this. The lexicalized HMM is formulated as follows:

Given a sequence of words W and a sequence of POS tags S. Compute T such that

$$\hat{T} = \arg\max_T P(T \| W, S) \tag{2.3}$$

After applying Bayes rule Equation 2.3 can be rewritten as follows

$$\hat{T} = \arg\max_T P(S \| T) P(W \| S, T) P(T) \tag{2.4}$$

In order to ensure the tractability of Equation 2.4, the following additional assumptions are made:

1. The current state is dependent on the previous state as well as the previous J words
2. The current word is dependent on the current state as well as the previous K POS tags
3. The current POS tag is dependent on the current state as well as the previous L words

After applying the above assumptions (J, K, and L =1), the final equation is written as follows:

$$\hat{T} = \arg\max_t = \prod_n P(S_n \| T_n, W_{n-1}) P(T_n \| T_{n-1}, W_{n-1}) P(W_n \| T_n, S_{n-1}) \tag{2.5}$$

2.2.1.2 Conditional Random Fields

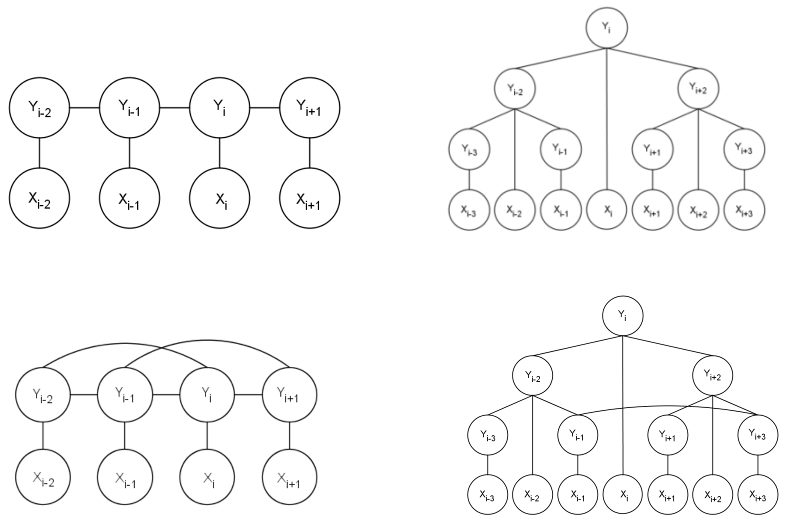
One of the disadvantages of hidden markov models is that they are not able to take into account overlapping features from the observed sequence. Conditional random fields (Lafferty, 2001) have been applied to various natural language processing tasks (Sha and Pereira, 2003; McCallum and Li, 2003; Ritter et al., 2011). A linear chain conditional random field is a graph G which represents conditional distribution of the labels Y given the evidence X (Li et al., 2010). The equation representing this distribution is written as follows (Lafferty, 2001):

$$P(Y\|X) = \frac{1}{Z(X)} e^{\sum_k \lambda_k f_k(y_t, y_{t-1}, x)} \tag{2.6}$$

Binary functions, known as feature functions, of form $f_i(y_i, y_{i-1}, x)$ are used to relate the evidence variables (W, S) to the hidden states (Sutton and McCallum, 2006). The feature functions are allowed to incorporate overlapping evidence from the entire sequence. The feature function weights (λ) are estimated from training data by maximizing the log likelihood of $\sum P(Y_i\|X_i)$ over all the training data via L-BFGS (Li et. al., 2010).

Some of the shortcomings of linear chain CRFs are that they are unable to model long range dependencies as well as the syntactic structure of text. Li et al. (2010) claim this information is very important for the task of opinion retrieval. They suggest that terms are often related to one another by conjunctions and different conjunctions can suggest changes in polarities. An example of this are the two conjunctions “and” and “but”; “and” will often lead to the same polarity and “but” will often lead to a change in polarity. In order to account for the long range dependencies, Li et al. (2010) propose a skip tree CRF.The skip-tree CRF is a combination of a skip-chain CRF (Sutton and McCallum, 2004) which allows for edges between non adjacent states and a tree CRF (Li et al., 2010) which adds a tree structure to the CRF. The added complexity of the above mentioned extensions to the linear chain CRF has a dramatic effect on complexity of inference. Due to the potential of complex loop structures being present in the graph, exact inference is intractable (Sutton and McCallum, 2004). Instead, approximate inference is performed using a method called tree re-parameterization (Wainwright et al., 2001).

Figure 2.1: Conditional Random Fields: (Top row) Linear Chain, Tree. (Bottom Row) Skip Chain, Skip Tree (Li et al., 2010)



2.2.2 Other Supervised Methods

Other supervised approaches involve the use of statistical classifiers such maximum entropy (Blair-Goldensohn et al., 2008) and support vector machines (SVM) (Yu et al., 2011). Blair-Goldensohn et al. (2008) propose a method for extracting ratable aspects from reviews of local services such as restaurants, local businesses and hotels. They observe that a large number of online searches for local services belong to a small number of categories (hotels and restaurants). Based on this observation, they train individual models for each of the high frequency domains. The models consist of a set of "static" aspects and a set of binary classifiers (1 for each aspect) used to determine if a sentence contains its corresponding aspect. The static aspect models are then combined with a frequency based dynamic aspect model to extract the aspects in the reviews.

2.2.3 Semi Supervised and Unsupervised Methods

2.2.3.1 Frequency based Methods

Product features and aspects are often made up of single phrases (Wu et al., 2009). A number of techniques for aspect extraction exploit this fact (Hu and Liu, 2004; Popescu et al., 2005; Blair-Goldensohn, 2008; Wu et al., 2009) by extracting frequent noun phrases as candidates and then filtering the candidates by applying a scoring mechanism. Hu and Liu (2004) apply association rule mining (Agrawal and Srikant, 1994) to extract all words and phrases that appear with high frequency. The extractions are then pruned based on "compactness" and "redundancy". They define a compact phrase as a phrase where at least 2 sentences contain the terms in the phrase and that the distance between any two adjacent phrase terms is less than 3. A redundant feature is a feature that appears in less than 3 sentences that do not contain a superset of the terms in the feature. Popescu et al. (2005) build upon the KnowItAll (Etzioni et al., 2004) system for extraction of pairs of related entities given a product domain. The KnowItAll system uses seed relations to generate patterns to extract candidates that participate in the relationships. It then uses point wise mutual information (PMI) (Turney, 2001) between the candidates and a set of generated indicator phrases as features for Naïve Bayes classifier to assign a probability to each of the candidates. In Opine (Popescu et al., 2005), high frequency noun phrase are selected as the candidates and are scored using meronym discriminators based on the product class. Language models have shown to be useful for filtering candidate noun phrases (Scaffidi et al., 2007; Wu et al., 2009). Scaffidi et al. (2007) suggest that aspect phrases are likely to appear more often in reviews than in a general corpus of English. A language model of general English text is used to filter unigram and bigram noun phrases that appear in general English text with high probability.

2.2.3.2 Pattern Based Methods

Related to frequency based methods are pattern based methods (Etzioni et al., 2005; Qiu et al., 2009; Zhang et al.,2010), in which patterns/rules are used to extract target phrases. Qui et al. (2009) present the technique of double propagation to extract both opinions and targets. Using a set of relational rules, targets are extracted given a set of opinion seeds. Another set of rules is then used to extract more opinions. This process continues until there are no new opinion terms or targets added. Zhang et al. (2010) improve upon the techniques presented in (Qui et al., 2009) by adding "part-whole" and "no" patterns to improve recall and a feature ranking mechanism based on feature importance to improve precision. They define "part-whole" relations as those that indicate that one of the participants is contained within the other; for example in the phrase "CPU of the computer", "CPU" is part of the "computer". The "no" pattern is defined as the term "no" followed by a noun phrase. In order to reduce the amount of noise added by increasing the number of patterns, Zhang et al. (2010) prune the candidates by filtering those of low "importance". Feature "importance" is calculated based on two assumptions:

1. Important candidates are those that have multiple opinions expressed about them and participate in many distinct "part -whole" and "no" relations.

2. Important features appear more frequently than less important features.

The first assumption is incorporated by formulating the problem in the HITS framework (Kleinberg, 1999). In HITS, documents are given a hub score and an authority score such that documents with high hub scores point to pages that have high authority scores. This relationship is formalized as follows:

Let G be a bipartite graph such that (i, j) exists in E if and only if there exists a link from document i to document j, and let $A(i)$ and $H(i)$ be the hub score and authority score for document i.

$$A(i) = \sum_{(j,i)\in E} H(j) \tag{2.7}$$

$$H(i) = \sum_{(i,j)\in E} A(j) \tag{2.8}$$

Power iteration can be used to determine a solution for $A(i)$ and $H(i)$ by putting the equations into matrix form as follows: Let A and H be column vectors such that $A_i = A(i)$

and $H_i = H(i)$ and let L be the adjacency matrix of graph G.

$$A = L^T H \tag{2.9}$$
$$H = LA \tag{2.10}$$

Zhang et al.,(2010) treat the features as the authorities and the indicators (opinion words, part whole relations) as the hubs. After running the HITS algorithm, the authority score for each feature is multiplied by the log of the frequency of the feature to produce the final importance score.

2.2.3.2.1 Dependency Parsing A number of pattern based methods construct patterns using the relationships between individual terms extracted from a dependency parse (Wu et al., 2009; Qui et al., 2009, Zhang et al., 2010). In a dependency parse, a sentence is represented as a tree where each node represents a term and the edges between each term represent the dependency relation between them. Dependency relations are defined as asymmetric binary relations between a head word and its modifier (Lin, 1998). In a typed dependency parse, (Lin 1998; Marfenne et al., 2006) labels representing the grammatical relationships such as subject and object are assigned to each of the edges.

Wu et al. (2009) propose a different type of dependency parse based on phrases instead of individual words. They claim that the word based dependency parses lose information contained in the constituent parse due to the lack of the major syntactic structures (noun phrases, verb phrases and prepositional phrases). To account for this they allow nodes in the dependency graph to be complete phrases. The dependency graph is computed from an existing word level dependency graph by merging nodes that belong to the same constituent phrase.

2.2.3.3 Clustering Based Methods

Clustering approaches, which include the work presented in this thesis, attempt to group noun phrases together based on their similarity. In (Raju et al., 2009; Du and Tan, 2009), group average agglomerative clustering is applied to a similarity matrix computed from a list of candidate noun phrases. Candidate noun phrases are extracted in a similar manner to those presented in the phrase based methods. Noun phrases are pruned based on their pointwise Kulback Leibler Divergence (KLD) (Tomokiyo et al., 2003) with a general English corpus. The similarity between two noun phrases is calculated as follows:

Let P_1 and P_2 be two noun phrases. Let S_1 and S_2 be the set of all unigrams and bigrams belonging to P_1 and P_2

$$Sim(P_1, P_2) = \frac{2\|S_1 \cup S_2\|}{\|S_1\| + \|S_2\|} \tag{2.11}$$

In group average agglomerative clustering, initially each candidate represents a single cluster. The most similar clusters are then iteratively merged together according to the average similarity between all the points in the candidate clusters.

$$AS(x) = \frac{KLD(x)}{AHD(x)} \tag{2.12}$$

After clustering, attribute names were extracted from the clusters by selecting the ngram with the highest attribute score (equation 2.12). Pointwise KLD is calculated as $KLD(x) = P(x)log\frac{P(x)}{Q(x)}$. P is the probability of ngram x in its cluster and Q is the probability of ngram x in the rest of the clusters. Average head distance (AHD) is defined as the average distance between the ngram and the right most word in the noun phrase.

Du and Tan (2009) build upon the information bottleneck algorithm proposed by Tishby et al. (1999) to simultaneously cluster both opinion words and aspects. In the information bottleneck algorithm, clusters of one random variable are joined together such that they minimize the change in mutual information between the clustering and the other random variable (Slonim and Tishby, 1999). The change in information gain caused by merging clusters C_i and C_j according to the following equations:

$$\sigma I(c_i, c_j) = (P(c_i) + P(c_j)) \times D_{js}\left[P(y_i\|c_i), P(y_i\|c_j)\right] \tag{2.13}$$

In Equation 2.13, D_{KL} is the Kulback Leibler divergence and D_{JS} is the Jenson-Shannon Divergence (Lin, 1991). In Du and Tan (2009), Equation 2.13 is calculated based on the co-occurrence between the feature terms and the opinion terms. This, they claim, loses much of the semantic information. In order to incorporate this information Du and Tan (2009) extend the information bottleneck algorithm by adding the semantic distance based on the Chinese lexicon HowNet[1] to Equation 2.13

2.2.3.4 Model Based Approaches

2.2.3.4.1 Classification Based Approaches Supervised machine learning has shown to work well for aspect extraction(Yu et al., 2011; Jin et al., 2009; Li et al., 2010; Blair-Goldensohn et al., 2008), however, annotated training sets are often too costly to create. In order to avoid this method seeds can be used to bootstrap a training set (Probst et al., 2007). In Probst et al.(2007), each word is labelled using a naïve Bayes classifier according to if whether it is either a product attribute or the attributes value.A set of seed attribute value pairs is used to bootstrap the Naive Bayes classifier using the co-EM algorithm (Ghani and Jones, 2002). In co-EM training, multiple classifiers are trained on different feature sets or “views” of the training data. The bootstrapped data set is used to train the first view classifier which is used to label the unlabeled corpus. This new data is then used to train the second view classifier which re-labels the corpus. Finally, the first view classifier is re-trained on the corpus. This process is continued until the classifiers converge (Probst et al., 2007).

2.2.3.4.2 Topic Modelling Approaches Topic Modelling has become a popular way to discover the hidden semantic structure found in documents (Lafferty and Blei, 2009). It is based on the concept that documents are mixtures of topics, which are distributions over the words. A process for generating documents can then be formulated as follows:

1. Select a random set of topics;
2. Select a topic z randomly from the topic set;
3. For each word w in the document d select a word from the topic.

[1]www.keenage.com

Two techniques for topic modeling are probabilistic latent semantic analysis (PLSA) also known as probabilistic latent semantic indexing (PLSI)(Hoffman, 1999) and latent dirichlet allocation (LDA)(Blei, 2003). In PLSA, the formulation above is formalized via the following model:

$$P(d,w) = P(d)P(w|d) \tag{2.14}$$

$$P(w|d) = \Sigma_z P(w|z)P(z|d) \tag{2.15}$$

PLSI falls short of being a complete generative model since it provides no generative model at the document level (Blei et al., 2003). It is therefore not able to apply a probability to an unseen document (Blei, 2003; Titov and Macdonald, 2008). LDA achieves a document level model by defining a distribution over the individual topics. The process for generating a document via LDA becomes

1. Select a topic distribution t from t Dir(α)
2. For each word in document d
    (a) Select a topic z from topic distribution t
    (b) Select a word from topic z

Unlike PLSA in which the maximum likelihood estimates for the model parameters can be computed via expectation maximization, exact inference under the LDA model is intractable (Blei et al., 2003). Instead, approximate inference methods such as vibrational EM (Blei et al., 2003) and Markov Chain Monte Carlo (MCMC) methods such as Gibbs sampling (Griffiths and Steyvers, 2004) have been developed to estimate the model parameters.

2.2.3.4.3 Multi grain Topic Modelling LDA and PLSA fall short due to the fact that they can only take into account document level co-occurrence. Aspects are often found in every review so therefore document level co-occurrence is not enough to identify them (Titov and Macdonald, 2008). In order to address the shortfalls of LDA and PLSA, Titov and McDonald, (2008) propose multigrain LDA. In multigrain LDA, words are sampled from one of two sets of topics: global topics and local topics. Global topics are assumed to be fixed for an entire document, where as local topics change across the document based the contexts of the words. Documents are represented as a collection of sliding windows containing an overlapping set of n sentences. Each of the windows w, share a common global topic distribution; however they have their own local topic distribution. The windows also have a preference distribution representing the preference of local topics versus global topics. Words are then sampled from either the distribution of global topics or a distribution of local topics. The generation process is as follows:

1. For each sentence s and document d let $t^{gl} \approx Dir(\alpha^{gl})\phi_{d,s}(w) \approx Dir(\gamma)$
2. For each window w let $t_w^{loc}\ Dir(\alpha^{loc}), c \approx Beta(\alpha^{mix})$
3. For each word i in document d
   (a) Select a window w_i from $\phi_{d,s}$
   (b) Select k from c
   (c) if k == global then select a global topic t_i from t^gl
   (d) if k == local then select a local topic t_i from t_w^{loc}
   (e) Select a word from topic t_i

Words can be sampled from any window as long as the sentence containing the word is contained within the window. We compare our method to multi-grain LDA in section 4.4.

A number of other techniques have applied topic modelling to the task of aspect extraction. Brody and Elhadad (2010) address the locality problem by treating each individual sentence as a document and use LDA to extract local topics relating to aspects.

The methods presented above address the issue of “local” vs. “global” concepts present in reviews, however they do not provide any mechanism for domain knowledge to be added to the model. Zhai et al. (2011) address this by allowing for “must-link” and “cannot-link” constraints (Andrzejewski, 2009; Zhai et al., 2011). The constraints are included in the

LDA model by multiplying the probability used to determine the topic for each word computed via the standard LDA model by the probabilities computed based on the constraints. The methods proposed by Zhai et al. (2011) assume that the aspect terms have already been extracted, therefore they rely on a pre-existing extraction mechanism. Mukherjee and Liu (2012) propose a semi supervised model for extraction and categorization. In their proposed model, each topic/aspect has a separate distribution over words and seed sets, and the seed sets have a distribution over the seed terms. A word is sampled by first sampling for a non-seed word and a seed set. If a seed set is selected, a seed term is then sampled.

2.3 Semantic Lexicon Construction

The works in semantic lexicon construction can be divided into two groups: corpus based methods and web based methods (Igo and Riloff, 2009). Corpus based methods, (Thelen and Riloff, 2002), are usually applied on corpus pertaining to a small set of domains in order to construct domain specific lexicons. On the other hand, web based methods, (Hearst, 1992; Snow et al., 2005), operate on the World Wide Web and often focus on creating or expanding large domain independent lexical resources such as WordNet (Miller, 1990). There has been a considerable amount of work done in the area of semantic class discovery and semantic lexicon creation. Similar to opinion and feature extraction, unsupervised techniques have been a major focus of study due to cost of developing an annotated corpus for training.

2.3.1 Term Similarity

There has been some work on applying term similarity measures to semantic class discovery. Riloff and Sheppard, (1997) suggest that terms belonging to the same semantic class often appear close together, for example “lions, tigers and bears” and “tuna fish”. Based on this claim, they propose a semi supervised method for generating semantic classes. Given a set of 5 seeds for each of the target classes (category terms), the algorithm extracts more terms that are deemed similar to the target classes. In the first step of the algorithm, all sentences containing at least one instance of the category terms are extracted. For each noun phrase such that the head is a category term, a window containing one noun phrase to the right and one noun phrase to the left is selected as the context window. Each word in the context window is then given a score based on Equation 2.16. The top 5 highest scoring words are added to the category words. The process is then repeated as many times as necessary. Riloff and Sheppard, (1997) suggest that the final set of terms should be manually judged by a human.

$$Score(W, C) = \frac{\mathit{Frequency\ of\ W\ in\ C\text{'}s\ context}}{\mathit{Frequency\ of\ W}} \tag{2.16}$$

Roark and Charniak (1998) claim that the techniques presented in (Riloff and Sheppard, 1997) favour low frequency nouns and therefore require a low frequency threshold (> 5). For candidate selection, Roark and Charniak, (1998) employ a similar scoring function to Equation 2.16. However, instead of using the same scoring function for both the final ranking and candidate selection, the final ranking is computed based on the log likelihood statistic described in (Dunning, 1993).

The scoring function presented in (Riloff and Sheppard, 2007) can be looked at as a simplification of pointwise mutual information (PMI)(Ahmadi,2012). PMI represents the amount of information the presence of one of the words gives about the presence of the other (Church and Hanks, 1990). It is computed as follows:

$$PMI(w_1, w_2) = \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \tag{2.17}$$

Turney (2001) proposes PMI-IR , a method for estimating PMI based on the number of web documents returned from a search engine given the terms as a query. Igo and Riloff, (2009) re-score candidate terms, presented by the Basilisk (Thelen and Riloff, 2002) system, by computing the PMI of the candidate with both the seed words and the target classes. Search queries are formed by combining the candidate terms and the targets with the AltaVista NEAR operator. The PMI scores are then computed using the methods presented in (Turney, 2001).

Lin (1998) defines the similarity between two words based on the dependency relationships in which the terms are participating. For each word Lin, (1998) generates a vector representing the frequencies f of the dependency relationships of the form (w, r, w'). The similarity between the two words is calculated as follows:

$$I(w, r, w') = log\frac{f(w, r, w') \times f(*, r*)}{f(w, r, *) \times f(*, r, w'} \tag{2.18}$$

Let $T(w)$ be the set of tuples (w', r), where r is a dependency relationship, such that $I(w, r, w') > 0$

$$Lin(w1, w2) = \frac{\Sigma_{(r,w)\in T(w1)\cap T(w2)} I(w1, r, w) + I(w, r, w2)}{\Sigma_{(r,w)\in T(w1)} I(w1, r, w) + \Sigma_{(r,w)\in T(w2)} I(w, r, w2)} \tag{2.19}$$

In (Lin,1998), similarity rankings are computed for each term, but there is no attempt to group similar terms into clusters. This has a number of drawbacks (Lin and Pantel, 2001). First, a global threshold is required in order to select the most similar terms. This can be both difficult to select, and any such threshold may not hold from term to term. Secondly, ranked lists of words do not represent coherent concepts (Lin and Pantel, 2001). This is due to the fact that for a given term, other terms from multiple contexts can be similar to it. Finally, the above distributional similarity measure is susceptible to error when presented with infrequent terms that have a small number of features.

2.3.2 Clustering Methods

A substantial number of works have applied clustering techniques to the task of lexicon and semantic class discovery (Caraballo, 1999; Lin and Pantel, 2001, Lin and Pantel, 2002). Caraballo (1999) applies agglomerative clustering in order to group similar nouns based on the number of times each noun appeared in a conjunction or appositive together. The similarity between two nouns n_1 and n_2 is calculated as follows:

Let V_k be a vector such that $V_k[i]$ = the number of times the ith noun appears in a conjunction or appositive with the kth noun.

$$cos(V_k, V_l) = \frac{V_k \dot{V_l}}{\|V_k\| \|V_l\|} \tag{2.20}$$

Clusters are merged together based on the weighted average of the similarity between the clusters contained within the candidates. For example, let cluster A be of size $\|A\|$ and cluster B be of size $\|B\|$. If cluster C is the result of merging clusters A and B, then the similarity between cluster D and cluster C is computed as the weighted average of the similarity between clusters A and D and cluster B and D.

Caraballo (1999) only models conjunction and appositive patterns and ignores all the other grammatical relationships. Lin and Pantel, (2001) use all dependency relationships as features. This leads to a large increase in the dimensionality of the vectors ($> 1,000,000$). In order to handle the large number of features Lin and Pantel (2001) employ a 2-step clustering process. In the first step, the data set is separated into subsets. A maximal clique algorithm is then used to find cliques for each term. They define a clique to be a set of words such that each word belongs to the top n similar words for every other term in the clique. In the second step, clique centroids are computed by averaging the feature vectors of the terms in the cliques returned from the first step. Finally, the cliques are merged together based on the similarity between their centroids. Lin and Pantel (2002) extend the work presented in (Lin and Pantel,2001). Similar to (Lin, 1998), each term is represented by a feature vector where each feature represents a grammatical relationship between the term and another word in the corpus. The value of each feature is computed as the pointwise mutual information between a context c and the word corresponding to the vector (Equation 2.21). The similarity between two words is calculated as the cosine similarity between the two feature vectors.

$$MI_{w,c} = \frac{\frac{F_c(w)}{N}}{\Sigma_i \frac{F_i(w)}{N} \times \Sigma_j \frac{F_c(j)}{N}} \tag{2.21}$$

The clustering process in (Lin and Pantel, 2002) is broken down into 3 phases. In phase 1, a list of the 10 most similar words is selected for each word based on the similarity matrix computed using equation 2.21. In phase 2, a set of tight clusters are selected as "committees". The committees are selected by first clustering the terms and then greedily selecting clusters such that the similarity of their centroids is below a pre-defined threshold. The process in phase 2 is repeated recursively until every term is similar to at least one of the committee centroids. The final set of committees represents the output clusters. Finally, in phase 3, each term is then assigned to its most similar cluster based on its similarity to the cluster's committee centroid.

2.3.2.1 Automatic Labeling of Clusterings

One of the major limitations to the clustering methods in the previous section is that they do not provide labels for the clusters. Pantel and Ravichandran, (2004) propose a method for addressing this using the vectors computed after the clustering process in (Lin and Pantel, 2002). The committee centroids are treated as "grammatical" templates representing the context of the term. The mutual information vectors scores of the terms that appear in the relationships are summed and ranked and the top scoring terms become the label of the cluster. Other approaches include (Caraballo, 1999) and (Staab, 2005). In (Caraballo, 1999), the class labels are derived from hypernyms are extracted out of the text using the 6 patterns proposed by Hearst (1992). In (Caraballo, 1999) and (Pantel and Ravichandran, 2004), the class labels are derived by relationships between the words that appear with the cluster members in the corpus. Often class labels do not co-occur with the cluster members. This is especially true in the case of online consumer reviews. For example, in the phrase "The pizza was delicious", there is no mention of the class label "food" for "pizza". Staab (2005) addresses this problem by using a web search engine to retrieve hypernyms.

2.3.3 Pattern Based Methods

Pattern based methods for semantic lexicon creation are often based on determining a set of lexico-syntactic patterns which represent the relationships between a term and a semantic class. One of the first techniques was that of (Hearst,1992), where a set of surface patterns were used to extract hyponym (is-a) relationships. The key intuition behind Hearst's method, is that if term x belongs to class Y, phrases similar to "Y such as X" must exist on the web. Hearst (1992) suggests 6 patterns (see table 3.4) for representing the hypnonymy relationships between terms in English. The patterns were derived using the following process:

1. Acquire a list of known relationship tuples, such as (Dallas, City);
2. Extract all sentences that contain both terms in the relationship;
3. Create a set of patterns based on the words that appear between the terms in the relationships.

A number of works have extended the work of Hearst,(1992). Berland and Charniak, (1999) apply a similar process to (Hearst, 1992) for discovering lexico-syntactic patterns which represent part-of relationships. Snow et al. (2005) and Girju et al. (2006) apply machine learning in order to automatically determine rules for extraction. In (Snow et al., 2005) pattern candidates are selected by extracting all nouns and the dependency paths between them. The candidate patterns are then split into two groups: known hypernym and known non-hypernym based on the WordNet (Millar, 1990) relationship between the nouns that led to its extraction. Snow et al., (2005) then train a statistical classifier to determine if a pair of words forms a hypernym − hyponym relationship. Girju et al. (2006) also apply machine learning for the purpose of discovering rules for determining relationships between terms; however, unlike (Snow et al. , 2005) they focus on meronymy relationships similar to those in (Berland and Charniak, 1999). Vechtomova and Robertson (2012) develop a technique for automatically discovering instances of the topics presented in the TREC related entity finding task. For each of the topics, a set of queries were constructed based on patterns presented in (Hearst, 1992). A set of documents were retrieved by passing the queries to a commercial search engine. Finally, the instances matching the patterns were extracted from the documents. The resulting instances were selected as seed entities belonging to the given category.

Pasca (2004) applies the pattern in Example 2.3.1 to extract categories and instances from the web. Ravichandran and Hovy (2002) propose a method to learn surface patterns for extracting answers to questions such as "When was X born?"

Example 2.3.1. Let X represent a category and N represent an instance

$$[startOfSent]\ X\ [suchas|including]\ N\ [and|,|.]$$

2.3.4 Bootstrapping Approaches

The process used in (Hearst, 1992) is similar to a bootstrapping approach, however in (Hearst, 1992) the final step involved a manual search through the extracted patterns. There has been a considerable amount of work applying bootstrapping methods to automatically generate extraction patterns (Brinn, 1998; Aigchtein and Gravano, 2000; Thelen and Riloff, 2009). Based on the DIPRE system (Brinn, 1998), Snowball (Aigchtein and Gravano, 2000) extracts tuples of the form (organization, location) from a large corpus. A set of patterns is first generated by finding all pairs of named entities of type LOCATION and ORGANIZATION that match a seed tuple. For each match, a 5-tuple containing the two entities and a set of term vectors representing the middle, left and right contexts is created. The left and right contexts are taken from a window of size n around the seeds and the middle vector is created from the text between the two seeds. The matches are then clustered based on the inner product between the middle, left and right term vectors. Finally, the new set of patterns is generated by computing the centroids of the middle, left and right vectors of each cluster.

Let T_1 be a tuple of the form $(l_1, t_1, m_1, t_2, r_1)$ and T_2 be a tuple of the form $(l_2, t_3, m_2, t_4, r_2)$

$$Match(T_1, T_2) = \begin{cases} l_1 l_2 + m_1 m_2 + r_1 r_2 & \text{if terms match} \\ 0 & \text{otherwise} \end{cases} \tag{2.22}$$

Once the new set of patterns is generated, they are used to expand the seed tuple set. For each pair of named entities of type LOCATION and ORGANIZATION a 5-tuple of the same form as those in the pattern generation step is created. The candidate tuples are compared to each of the patterns using Equation 2.22. In order to prevent drift caused by erroneous extractions during the bootstrapping process, both the patterns and the extracted tuples are given a confidence score. The tuples generated by each pattern are compared to previous high confidence extractions. If the location and organizations match then the tuple is considered a positive match, otherwise it is considered a negative match. The confidence for a pattern P is then computed as follows:

$$Conf(P) = \frac{\#\ of\ Positive\ Matches}{Total\ \#\ of\ Matches} \tag{2.23}$$

A tuple's confidence is computed from the confidences of the patterns that led to its extraction. In the extraction step, each extraction was given a score based on how similar its context vectors were to the patterns context vectors. The scores are then combined to compute the final tuple confidence using the following equation:

Let P be the set of patterns that generated T and let C be the context associated with T

$$Conf(T) = 1 - \prod_i 1 - (Conf(P_i) \times Match(C_i, P_i)) \tag{2.24}$$

Thelen and Riloff (2002) propose Basilisk, a method for generating semantic lexicons based on automatically extracted patterns. In the first step of Basilisk, extraction patterns are generated for each noun phrase by running AutoSlog (Riloff, 1996) over the corpus. In AutoSlog, extraction patterns are generated for each noun phrase based on a set of heuristic rules made up of grammatical constructs. (See table 2.1). The patterns are then scored based on their tendency to extract seed terms using the following equation:

Let F_i be the number of known category terms extracted by pattern i and let N_i be the number of nouns extracted by pattern i

$$RlogF(pattern_i) = \frac{F_i}{N_i} log_2 F_i \tag{2.25}$$

The highest scoring patterns are selected to be part of the pattern set used in the subsequent steps of the process.

The noun phrases extracted by the pattern pool are selected as lexicon candidates.

Table 2.1: Example Heuristics used by AutoSlog (Riloff,1996)

Heuristic Rule	Example
$< subj >$ passive-verb	$< car >$ was fixed
$< subj >$ active-verb	$< person >$ walked
$< subj >$ verb infln	$< person >$ attempted to fix
$< subj >$ aux noun	$< person >$ was victim
active-verb prep $< np >$	hit with $< object >$
passive-verb prep $< np >$	was aimed at $< target >$

Each candidate term is scored based on the number of patterns, including those not in the pool, that led to its extraction and the number of known lexicon terms extracted by the patterns. In order to prevent patterns with high numbers of known term extractions from skewing the candidate terms scores, the average logarithm of the known term frequencies is used in the calculation of the term score. The final candidate term score is computed as follows:

Let P_i be the number of patterns that extract word i

$$AvgLog(word_i) = \Sigma \frac{log_2 F_{j+1}}{P_i} \tag{2.26}$$

The highest ranking terms according to Equation 2.26 are then added to the lexicon and the bootstrapping process is repeated.

Chapter 3

Methodology

Lin (1997) suggests that terms that appear in similar contexts have similar meanings. This relationship is even more evident when it comes to product features/aspects and the subjective modifiers of these aspects. For example people often use different words when referring to food than when they are referring to the wait staff. It would be very common for someone to say that the "wait staff" was "polite", but it is very unlikely that someone would refer to their "meal" as "polite". This suggests that by grouping terms together based on the similarity of their contexts one could build lexicons of semantically related terms. Based on this premise we propose the following algorithm:

1. Apply dependency parsing to a corpus in order to retrieve typed dependency triples
2. Extract candidate nouns from the corpus
3. Cluster each candidate where the cluster similarity is the average Lin's similarity (Lin,1998) between the terms in the clusters
4. Extract hypernyms from the web for each noun candidate
5. Merge clusters based on the hypernyms that appear in the cluster

Before beginning our clustering process, we first process the entire corpus using typed dependency parsers. As described in Section 2.2.3.2.1, a typed dependency parse describes

Table 3.1: Example Dependency Triples with Part of Speech

Sentence: The staff, however, is dismal
det NN DT(staff-2,The-1)
nsubj JJ NN(dismal-7,staff-2)
advmod JJ RB(dismal-7,however-4)
cop JJ VBZ(dismal-7,is-6)

the grammatical relationships between individual terms in a sentence. More specifically, the parse gives information about the relationships between the head of a noun phrase and the terms that modify it (Wu, 2009; Marfenne, 2006). The modifiers and their relationships to a given head word can provide a local context for a terms use in a document (Lin, 1997). For this purpose, we process each of the reviews using the Stanford dependency parser. The typed dependency parse returns tuples of the form $(w1, r, w2)$ where $w1$ and $w2$ represent terms in the corpus and r represents the grammatical relationship between the terms. We append the part of speech tags for each of these words to each of the dependency tuples to form a complete relationship between each term in the corpus. We represent the context for each term as the set of relationships for which it is a participant.

3.1 Candidate List Generation

Once each term has been given a context, we compute the candidate terms for the aspects. Aspects can be defined as any feature/attribute of a product; for example "battery life" for mobile phones or "food quality" for restaurants. We assume that each noun can be considered a valid candidate for extraction; therefore all of the terms, in the corpus, that had been tagged with an NN, NNS, NNP or NNPS are selected as candidates.

In order to reduce the effect of errors introduced by the part of speech tagger we apply two filters to the candidate nouns. For the first filter, we remove all candidates that appear as a noun less than 50% of the time. For the second filter, we assume that the subjective nature of the corpus suggests that the important aspects are the terms that are likely to have subjective modifiers; therefore, we remove terms that do not occur with at least one "modifier" dependency (see Table 3.2).

Table 3.2: Modifier Dependency Relations

Term POS (Part of Speech)	Dependency Relationship	Modifier POS
JJ,JJR,JJS	amod,rcmod	NN,NNS,NNP,NNPS
NN,NNS,NNP,NNPS	nsubj	JJ,JJR,JJS

3.2 Clustering

3.2.1 Lin's Similarity

As stated earlier, Lin suggests that terms that appear in the same local context have similar meanings (Lin, 1998). He defines the similarity between two words as the amount of information in common between the words, divided by the information contained in the description of each of the words individually. The description of a word is defined as the complete set of dependency tuples containing the word of the form $(w, *, *)$, and the information in common between two words is defined as the information from all the tuples that appear in the descriptions of both of the words (Lin, 1998)

$$I(w, r, w') = log\frac{f(w, r, w') \times f(*, r, *)}{f(w, r, *) \times f(*, r, w')} \tag{3.1}$$

Let $T(w)$ be the set of tuples (w', r), where r is a dependency relationship, such that $I(w, r, w') > 0$

$$Sim(w1, w2) = \frac{\Sigma_{(r,w)\in T(w1)\cap T(w2)} I(w1, r, w) + I(w, r, w2)}{\Sigma_{(r,w)\in T(w1)} I(w1, r, w) + \Sigma_{(r,w)\in T(w2)} I(w, r, w2)} \tag{3.2}$$

We define f(w,r,w') as the total number of dependency tuples of the form (w,r,w') .Whenever the (*) appears it is used to denote all tuples that match the relationship pattern defined by the other terms. For example, f (good, a mod,*) would refer to the total number of tuples where "good" has the relationship "amod". The following example describes how Lin's similarity is calculated for the terms "pizza" and "fries"

Example 3.2.1. Lin's similarity between "pizza" and "fries"

Dependency Relations for “pizza” and “fries”

Fries	(crispy, fries), amod (fries, delicious)
Pizza	nsubj(good, pizza), amod(pizza,tasty), amod(pizza, delicious)

Frequency Counts for Example Dependencies

Relation	Frequency
pizza,amod,delicious	4
pizza,amod,tasty	2
pizza,nsubj,good	2
fries,nsubj,crispy	6
fries,amod,delicious	2

$$\begin{aligned}
I(pizza, amod, delicious) &= \log \frac{4 \times 8}{6 \times 6} \\
&= \log \frac{8}{9} \\
I(pizza, amod, tasty) &= \log \frac{2 \times 8}{6 \times 2} \\
&= \log \frac{4}{3} \\
I(good, nsubj, pizza) &= \log \frac{2 \times 8}{2 \times 2} \\
&= \log 4 \\
I(fries, amod, delicious) &= \log \frac{2 \times 8}{2 \times 6} \\
&= \log \frac{4}{3} \\
I(crispy, nsubj, fries) &= \log \frac{6 \times 8}{6 \times 6} \\
&= \log 8
\end{aligned}$$

$$\begin{aligned}
Desc(pizza) &= I(pizza, amod, delicious) + I(good, nsubj, pizza) + I(pizza, amod, tasty) \\
&= \log\frac{8}{9} + \log\frac{4}{3} + \log 4 \\
Desc(fries) &= I(fries, amod, delicious) + I(crispy, nsubj, fries) \\
&= \log\frac{4}{3} + \log 8 \\
Com(fries, pizza) &= I(fries, amod, delicious) + I(pizza, amod, delicious) \\
&= \log\frac{8}{9} + \log\frac{4}{3}
\end{aligned}$$

$$\begin{aligned}
Lin(pizza, delicious) &= \frac{Com(fries, pizza)}{Desc(fries) + Desc(pizza)} \\
&= \frac{\log\frac{8}{9} + \log\frac{4}{3}}{\log\frac{4}{3} + \log 8 + \log\frac{8}{9} + \log\frac{4}{3} + \log 4}
\end{aligned}$$

3.2.2 Hierarchical Clustering

Once the candidate terms have been selected, we apply agglomerative hierarchical clustering where for our initial clustering each individual term represents a single cluster. Following each step of the clustering process, the cluster pair with the maximum similarity, according to Equation 3.3, is merged into a single cluster. The clustering process is continued until the maximum cluster similarity has fallen below a stopping threshold. A diagram of the clustering process is shown in Figure 3.1.

Let C_1 and C_2 be disjoint clusters

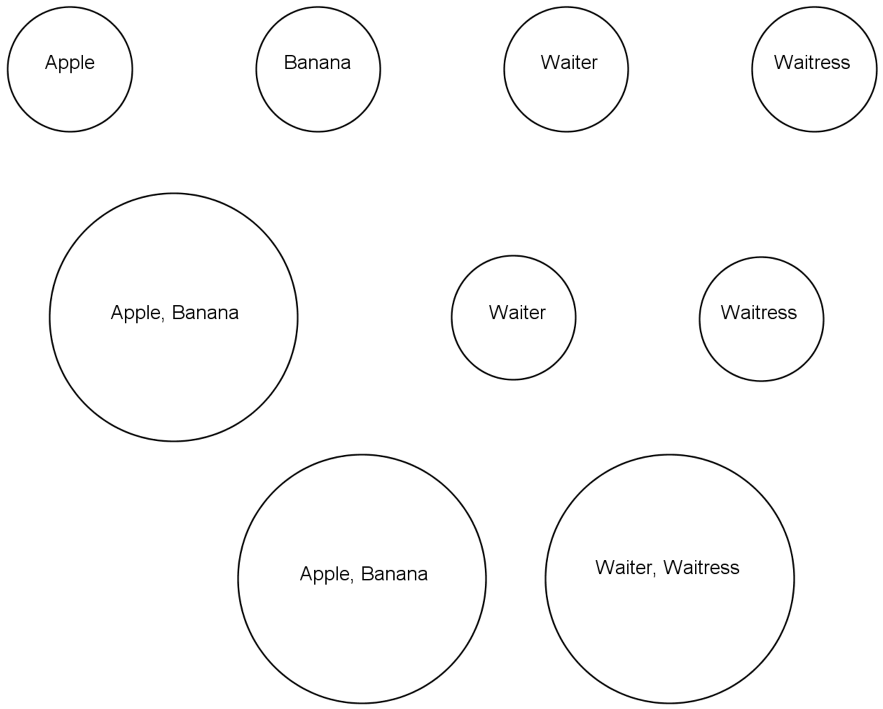
$$Sim(C_1, C_2) = \frac{1}{\|C_1\| \times \|C_2\|} \Sigma_{c_1 \in C_1} \Sigma_{c2 \in C_2} Sim(c_1, c_2) \tag{3.3}$$

For efficiency purposes we compute a similarity matrix S before initiating the clustering process. The matrix is constructed such that $S_{ij} = Sim(Ci, Cj)$ where C_i indicates the *ith* cluster and $Sim(C_i, C_j)$ refers to Equation 3.3

Table 3.3: Example Similarity Matrix

	Apple	Banana	Waiter	Waitress
Apple		0.5	0.01	0.02
Banana	0.5		0.03	0.01
Waiter	0.01	0.03		0.4
Waitress	0.02	0.01	0.04	

Figure 3.1: Clustering process for the similarity matrix in Table 3.3. Each row represents a round of clustering



3.3 Merging Clusters based on Hypernyms

3.3.1 Motivation

Our initial clustering results in a large number of fine grained semantic classes; however, what is often required in many applications, are higher level classes. This is more prominent in review systems where product features are broken down into high level categories such as food, service and environment. In order to address this, we propose a system for merging the lower order classes based on the hypernyms of the terms in each cluster. Our algorithm proceeds as follows:

3.3.1.0.1 Algorithm

1. Compute hypernym extraction patterns using the terms in the candidate list;
2. Submit extraction patterns as search queries to a search engine;
3. Extract noun phrases using the extraction patterns;
4. Merge clusters based on similarity of the extracted hypernyms.

3.3.2 Generation of Extraction Patterns

The first step in our algorithm is to generate a series of patterns for extracting hypernyms of each of our terms in the candidate set. We begin by generating extraction rules based on Hearst's lexico-syntactic patterns (Hearst, 1992) for each of the terms in the candidate set. The rules consist of 6 patterns which have been shown to be effective for extracting hyponyms. Since we are interested in hypernyms we adapt the rules to allow for hypernym extraction by replacing the NP_Y with the target terms. An example of this is shown in Table 3.4.

Table 3.4: Example of adapting Hearst's rules for term "pizza"

Pattern	Adapted Pattern
NP_X and other NP_Y	pizza and other NP_Y
NP_X or other NP_Y	pizza or other NP_Y
NP_X or other NP_Y	pizza or other NP_Y
NP_Y such as NP_X	NP_Y such as pizza
Such NP_Y as NP_X	Such NP_Y as pizza
NP_Y including NP_X	NP_Y including pizza
NP_Y , especially NP_X	NP_Y , especially pizza

3.3.3 Extraction of Hypernym Candidates

Each of the generated patterns is used to form the basis for a search query into a commercial search engine similar to the method proposed in (Vechtomova and Robertson, 2012). Many of the terms in our candidate set have multiple meanings depending on the context of use; for example "chair" could mean furniture or a position on a company's board of directors. Ignoring this fact , leads to a large number of irrelevant hypernyms being extracted for these candidates. In order to address this, we append the queries with a word representing the context or domain of the corpus, in our case "restaurants". We believe that using the name of the "domain" does not change the level of supervision since any application developer would know what type of product the reviews are about or could be determined automatically.

After removing HTML tags from the documents, we process each of the documents using a shallow parser (Illinois Shallow Parser) (Punyakanok and Roth, 2001) in order to identify the noun phrases contained in them. For each of the generated extraction patterns (see Table 3.4), we create a regular expression pattern which is used to extract out the noun in the phrase which is the closest in proximity to the target term.

3.3.4 Merging of Clusters based on Hypernyms

After extracting the hypernym candidates, we compute the hypernym representation (see Table 3.5) for each of the clusters. Each cluster has its terms replaced by the hypernyms

Table 3.5: Clusters and their Top 5 Ranked Hypernyms

Cluster Terms	Top 5 Hypernyms
Carmel, caramel, nut, pumpkin, banana apple, coconut, mango, lemon lime, cherry, peach, strawberry, berry buttermilk, truffle, rum	Flavors, drinks, desserts, dessert, dishes
Fireplace, fire, oven, wood , coal, charcoal	Sources, elements, tools, fuels, risks
Pancakes, muffins, cookies, breads, meatballs pastries cakes, pies, biscuits, donuts, doughnuts, cupcakes, brownies, waffles, sticks, tarts, wraps slices, fruits, pieces, veggies, cravings, oysters ...	Foods, food, fare, items, dishes

that represent the term. The unique set of hypernyms is selected as the hypernym representation of the cluster. Within each cluster we rank each hypernym according to Equation 3.4

Let c be a cluster and let h be a hypernym in the hypernym representation of c, Let N be the total number of clusters and let n_h be the total number of clusters containing h in its hypernym representation

$$\begin{aligned} Score_c(h) &= TF_c(h) * IDF(h) \\ TF_c(h) &= \textit{\# of terms in cluster c with hypernym h} \\ IDF(h) &= \log \frac{N}{n_h} \end{aligned} \tag{3.4}$$

In order to reduce the effect of invalid hypernyms caused by errors in both the initial clustering and the hypernym extraction phase, hypernyms that appear with only a single element in the cluster are removed from the cluster representations. Clusters are then selected for merging according to their similarity. After each iteration, of the algorithm, we compute the similarity for each cluster pair. The pair with the maximum cluster similarity is selected as a candidate for merging. Since we do not have dependency information regarding the hypernyms, we cannot use Lin's similarity. Instead, we treat each cluster as a vector where the *ith* index represents the weight for the *ith* hypernym in the total set of hypernyms. We calculate the similarity between two clusters as follows:

Let $W(w)$ be the weight given to word w

$$CosineSim(C_1, C_2) = \frac{\Sigma_i c_{1i} c_{2i}}{\Sigma_i c_{1i} \Sigma_i c_{2i}} \tag{3.5}$$

$$SetOverlap(C_1, C_2) = \frac{\Sigma_{w \in C_1 \cap C_2} W(w)}{\min\left(\Sigma_{w \in C_1} W(w), \Sigma_{w \in C_2} W(w)\right)} \tag{3.6}$$

Both of the similarity measures, described in Equations 3.5 and 3.6, result in a local score and therefore allow clusters containing only common, low importance terms to be merged. To address this problem we compute the global maximum weighted cluster overlap and compare the maximum score at each round to it using Equation 3.7. We set two thresholds: one for the round score and one for the global maximum score in order to determine whether or not a cluster pair should be merged.

$$ClusterOverlap(C_1, C_2) = \Sigma_{w \in C_1 \cap C_2} min\left(F_{C_1}(w), F_{C_2}(w)\right) \tag{3.7}$$

$$\frac{ClusterOverlap(C_1, C_2)}{max_i CurrentClusterOverlap_i} > threshold \tag{3.8}$$

3.4 Aspect Oriented Sentiment

User opinions on various product aspects found in reviews can vary from the overall sentiment of the review. An example of this is in a restaurant review: a negative review may contain positive comments on the food, but also negative comments on both the physical environment and the service. This suggests that it is important to measure sentiment with respect to the individual aspects found within a review. To show how our techniques can be used to perform this task, we applied our methods to the task of determining the sentiment expressed with respect to food related terms present in a corpus of restaurant reviews. We define the target aspect(s) as the aspects for which we want to determine the sentiment. In the case of our experiment, the target aspect was “food”. Furthermore, we define the target review as the review for which we are measuring the sentiment. This differs from the corpus reviews which are the reviews we apply our clustering on.

Table 3.6: List of seed terms for food related terms

food dishes ingredients drinks vegetables salads seafood sides meats fruits appetizers

3.4.0.0.2 Algorithm

1. Apply term clustering on offline review corpus;
2. Generate a set of seeds to represent the target aspect(s);
3. Apply seeds on hypernym representation of clusters to build aspect lexicon;
4. Extract modifiers of the terms in the lexicon that appear in a review;
5. Score each modifier according to its polarity;
6. Aggregate polarity over all extracted modifiers and compute the overall score for the target aspect.

3.4.1 Lexicon Creation

In order to generate a lexicon, we assign a label to each of our clusters based on its hypernym representation (see Table 3.5). The hypernyms representing an individual cluster are sorted according to their score given by equation 3.4. The set containing the top K scoring hypernyms is then selected to act as the label of the cluster. Once each cluster has been assigned a label, a set of seeds is intersected with each of cluster labels. The terms belonging to the clusters for which the seed – label intersection is non-empty are selected to be part of the lexicon. For the purpose of simplicity, we use a small set of hand selected words related to the target aspect as the seeds. The complete list of seeds can be seen in table 3.6

3.4.2 Modifier Extraction

After creating the lexicon of aspect related terms, we take each term in the lexicon and extract all modifiers, from the typed dependency parse of the reviews, for which the lexicon term is the head. We extract the modifiers from the modifier relations described in the Section 3.1.

3.4.3 Aspect Polarity Scoring

We score each aspect according to the average polarity associated with modifiers that act on the aspect terms. Each modifier is given a score of +1 if it is a positive modifier, -1 if it is a negative modifier or 0 if it is an objective modifier. Then the average polarity is computed by summing over all modifier polarity scores and dividing by the total number of aspect related terms found in the review. Aspect terms that do not have associated modifiers are left out of the divisor; we feel that this is appropriate since these terms likely do not contribute to the overall opinion of the review.

3.4.4 SentiWordNet

Scores for each term were derived from SentiWordNet (Esuli and Sebastian, 2006), an ontology based on Princeton's WordNet. In WordNet (Miller, 1990), words with the same meaning are grouped together into sets called synsets. SentiWordNet associates polarity scores (positive, negative, objective) with each of the synsets contained within WordNet. In order to use SentiWordNet for polarity scoring, one must account for the fact that a word may belong to multiple synsets given its part of speech. Verma and Bhattacharyya (2009) suggest 3 ways, in which a single score can be computed from the scores for each synset that a term belongs to.

The first method is to take the maximum of the polarity scores for an each individual synset and then to compute the average over all synset. In the following equation, n is the total number of synsets for which W is a member, $Pos_k(W)$ is the positive score of the word for the kth synset and $Neg_k(W)$ is its corresponding negative weight score.

$$Score(W) = \frac{1}{n}\Sigma_k \max(Pos_k(W_k), Neg_k(W_k), Obj_k(W_k)) \tag{3.9}$$

The second method is to compute the maximum of maximums over all the synsets; that is for each synset compute the maximum of the positive and negative scores and then compute the maximum over all the synsets.

$$Score(W) = \max_k[\max_k(Pos_k(W_k), Neg_k(W_k), Obj(W_k)] \tag{3.10}$$

The final method is to compute the weighted average of the maximum of the positive and negative polarity. WordNet orders each term in a synset according to the number of times the term is used in the context given by the synset. Each term can then be given a weight based on where it occurs in the synset.

$$Score(W) = \frac{1}{n}\Sigma_k F_k(W_k) * \max(Pos_k(W_k), Neg_k(W_k), Obj(W_k)) \tag{3.11}$$

$$F_k(W) = \frac{1 - \mathit{Position\ of\ W\ in\ the\ kth\ synset}}{\#\ \mathit{of\ words\ in\ the\ kth\ synset}} \tag{3.12}$$

In order to take into account negations, we look for negation relationships ("neg") containing the modifiers and we flip the sign of the polarity score. Finally, we compute the overall aspect score by averaging over all the individual polarity scores of the modifiers.

3.4.5 Multiword Units

The opinions and aspects found in user reviews are often present as multi word phrases. Many of these phrases have a complex structure such as dish names consisting of multiple food ingredients other dishes (i.e pasta with tomato sauce, olive oil and chicken) (Vechtomova, 2013). In order to account for this we apply the technique proposed in (Vechtomova, 2013) to extract multi word phrases for both the modifiers and the dish names. We begin the process by parsing each sentence using a typed dependency parser such as the Stanford Parser (Marfenne, 2006). For multi word aspects, we first identify the single nouns from the aspect lexicon where each single noun is a head word in a syntactic dependency relation. Finally, we iteratively merge adjacent terms based on the following set of rules:

1. Adjacent terms that share dependencies: nn (noun modifier) , amod (adjective modifier) such as “field” in “field grass” and “blue” in “blue cheese” are merged. If the adjacent terms share pos (Possessive modifier) such as “church’s chicken” the phrase is merged if the normalized pointwise mutual information (NPMI) is greater than a threshold α.

2. Adjacent terms that share preposition or conjunction dependencies are merged if the NPMI is greater than a threshold β. We define adjacent to mean two terms such that the in-between words contain only determiners and words contained in the dependency relation, for example “of” in prep_of.

In both steps of the above process, we calculate normalized point wise mutual information (NPMI) (Bouma, 2009) to determine if a word should be merged to the phrase. Pointwise mutual information (PMI) (Church and Hanks, 1990) is a popular method for calculating co-occurrence based similarity (see Section 2.3.1). Unfortunately, it has no upper bound, which makes it difficult to apply a threshold. Unlike its unormalized counterpart, NPMI is bounded between 1 (two terms always occur together) and -1 (two terms never occur together). It is calculated as follows:

Let $P(X,Y)$ be the probability that Y appears immediately following X in the corpus, then

$$NPMI(X,Y) = \frac{log\frac{P(X,Y)}{P(X)P(Y)}}{-logP(X,Y)} \tag{3.13}$$

We use a similar process for extracting multiword modifiers to (Vechtomova,2013). First, we identify the modifiers d that appear in the dependency relations with the aspect nouns. We then merge the adjacent terms using the following set of rules:

1. If the modifier shares an “amod” or “rcmod” relationship with the aspect, then for all dependants c of modifier d

   (a) If c and d share a negation modifier dependency (neg) and the previous word $(d-1)$ is a verb (VB) or modal (MD) then add the previous word to the MWU
   (b) If c and d share an adverbial modifier dependency (advmod) then add both c and d to the MWU
   (c) All words that lie in between the beginning of the MWU and the end of the MWU are added so that the MWU is contiguous.

2. If the modifier shares a "nsubj" relationship with the aspect , then for all dependents c of d
   (a) Follow part a. of 1.
   (b) Follow part b. of 1.
   (c) If c and d share a prep_than relation and d has part of speech JJR, then add c to the MWU
   (d) Follow part c. of 1.

3. If the modifier shares a "dobj" relationship with the aspect, then for all dependents c of d
   (a) Follow part a. of 1.
   (b) If c and d share an "nsubj" relationship and d is either "I" or "we", then follow part c. of 1

After each rule is applied, we check to see if the aspect term is contained in the MWU, and if it is, we take the single unit modifier as the MWU. We found that the multiword modifiers were often not found in SentiWordNet. In order to address this, we use the lemma returned by the Stanford Core NLP system[1].

[1]The Stanford Natural Language Processing Group: http://nlp.stanford.edu/

Chapter 4

Results and Discussion

4.1 Experimental Setup

We evaluated our methods using two datasets presented in (Vechtomova, 2013). The first dataset, henceforth referred to as Testset 1, consisted of 157,865 English restaurant reviews taken from a major commercial review database representing 38,782 restaurants located in North America. The reviews were pre-processed to remove html tags and encodings (Ahmadi, 2012). In order to evaluate the clustering, we had four annotators label each of the cluster candidates (after filtering and removal of misspellings) with one of the following labels:

1. Food
2. Ambience
3. Physical Environment
4. Service (Waiter, Staff)
5. Service Attributes
6. Selection/Menu

7. Clientele
8. Value

The final annotated set was created by taking all annotations where at least 3 of the 4 annotators agreed. All other annotations were discarded. We found that 3 out of 4 annotators agreed on the class label for 69% of the candidates.

For our second dataset (Testset 2) we used the annotated set presented in (Vechtomova, 2013). The set contained a random selection of 600 reviews taken from the review corpus. Two annotators manually assigned labels for the phrases in the reviews. Each annotator labeled a non-overlapping set of 300 reviews and a third annotator went through each of the annotations and made corrections. The labels assigned were as follows:

1. Food /Dish
2. Positive Modifier (Phrases that modify dishes or aspects in a positive manner)
3. Negative Modifier (Phrases that modify dishes or aspects in a negative manner)
4. Aspect

4.1.1 Preprocessing

We found that our reviews contained a large number of spelling mistakes. To account for this we compared each word to a large list of English words. Candidates with a levenstein distance of less than 3 from a word in the list of English words were removed.

4.2 Evaluation

4.2.1 Clustering Without Hypernyms

For the purpose of evaluation we had two baselines: no clustering (all singleton clusters) and complete clustering (all clusters put in one group). We then evaluated our clustering method with stopping thresholds at cut-off points of size 0.02. The cut-off points were in the range starting at 0.02 and ending at 0.10. Due to the large number of non-entity candidates, we expected that our clustering would result in a large number of singleton classes, therefore, we evaluated our methods with and without removing singleton clusters.

In order to compare the performance of each of the clusterings, we calculated the F_1 score. F measures are a popular method for evaluating hierarchical clustering (Steinbach et al, 2000; Beil et al, 2002). The measure is computed by first calculating the F measure for each cluster using the following equation:

Let C be a clustering and G be a gold clustering

$$F_\beta(c_i, g_i) = (1+\beta^2)\frac{Precision(c_i, g_i)Recall(c_i, g_i)}{B^2 Precision(c_i, g_i) + Recall(c_i, g_i)} \tag{4.1}$$

$$Precision(c_i, g_i) = \frac{P(c_i, g_i)}{P(c_i)} \tag{4.2}$$

$$Recall(c_i, g_i) = \frac{P(c_i, g_i)}{P(g_i)} \tag{4.3}$$

We, then, aggregate the scores for the individual clusters as follows (Whissel, 2012)

$$FQ(C, G) = \sum_{g_i \in G} P(g_i) max_{c_i \in C} F_\beta(c_i, g_i) \tag{4.4}$$

A number of terms could not be clustered since their maximum Lin's similarity with every other term was zero. To evaluate the effect of these terms, we conducted two experimental runs. For the first run we included these terms in our gold set and kept them as singletons in our results and for the second run we removed the terms from both the results and the gold set. Tables 4.1 and 4.2 summarize the result of our experiments at each of the cut-off points

Tables 4.1 and 4.2 both show that there was a considerable increase in the F_1 score after applying our clustering technique. The large score from the second baseline (threshold of 0) was likely caused by the large percentage of "food" related terms relative to the other

Table 4.1: Cluster F_1 scores including non-cluster terms

Threshold	Singletons	0	0.02	0.04	0.06	0.08	0.10
Cluster F_1	0.025	0.244	0.330	0.207	0.153	0.159	0.122
# of Clusters		1	943	1209	1556	1764	1903

Table 4.2: Cluster F_1 scores excluding non-cluster terms

Threshold	Singletons	0	0.02	0.04	0.06	0.08	0.10
Cluster F1	0.035	0.218	0.418	0.276	0.208	0.159	0.122
# of Clusters		1	432	779	1045	1253	1392

aspects. Out of the 600 terms which were given a label belonging to an aspect, 431 of these were labeled as food.

Another observation is that our clustering methods resulted in a large number of clusters that were small in size. For a stopping threshold of 0.02, 60% of the resulting clusters had less than 3 members and 20% of them consisted of only a single member. This was likely caused by the large number of non-entity candidates. We define non-entity terms as terms that did not belong to one of the classes described in Section 4.1. On average, our annotators found that 60% of the candidates did not belong to any of the classes.

Another source of error that we found is that our methods had difficulty with dish terms that were also used to describe entire genres or types of restaurants. These include terms such as sushi, pizza, salad etc. These terms were clustered into the same clusters which contained words related to the physical properties of a restaurant such as place, bar and restaurant.

4.2.2 Clustering With Hypernyms

We evaluated three methods presented in Section 3.3.4 for merging the clusters using the hypernym representations. For our evaluation set, we used the same annotated set used in the previous experiment (without the non-cluster terms). For each method we evaluated the merging thresholds at cut-offs lying on a 2 dimensional grid with the cut-offs for each threshold differing by 0.1. The threshold cut-offs were in the range of 0 to 1 for both of

Table 4.3: Maximum F_1 score for various stopping thresholds (including non-cluster terms)

Similarity Method	0.02	0.04	0.06	0.08	0.10
Cosine Similarity	0.429	0.447	0.301	0.278	0.245
Weighted Cluster Overlap (IDF Weights)	0.429	0.297	0.237	0.172	0.120
Cosine Similarity (IDF weights)	0.428	0.378	0.240	0.162	0.137

Table 4.4: Maximum F_1 score for various stopping thresholds (excluding non-cluster terms)

Similarity Method	0.02	0.04	0.06	0.08	0.10
Cosine Similarity	0.521	0.559	0.394	0.369	0.328
Weighted Cluster Overlap (IDF Weights)	0.521	0.386	0.316	0.234	0.166
Cosine Similarity (IDF weights)	0.521	0.476	0.320	0.222	0.188

the thresholds. Tables 4.3 and 4.4 summarize the maximum F_1 measures for each of the cut-offs.

The results of our experiment clearly demonstrate an improvement in cluster F_1 after clustering based on the hypernyms. The maximum F1 score for the cut-off of 0.02 resulted in a (30%) increase in overall F_1 score as compared to the same cut-off in Table 4.1. The overall maximum F_1 was found at 0.04 using cosine similarity and no IDF weights. This differed from the non hypernym clustering which had its maximum F-score at a stopping threshold of 0.02.The performance of all three methods on our dataset was identical for the stopping threshold of 0.02, however, this was likely due to the large number of food terms in the dataset. All three methods resulted in the merging of large food clusters. The clusters relating to the other types had already been formed during the initial clustering phase.

4.2.3 Cluster Labeling

Since our cluster candidates consisted of a larger number of food items (431 out of 600) we chose to evaluate the cluster labeling on dish names. We evaluated our lexicon on two different annotated sets. The first set of annotations was from the items labelled as food used in the previous section for evaluating our clustering method. The second set

Table 4.5: Precision and Recall for Automatically Generated Food Lexicon

Annotation Set	Precision	Recall
Set 1	0.541	0.835
Set 2	0.591	0.813

was created by first extracting phrases that were annotated as food, as well as the word "food", itself from the set of 600 annotated reviews. Since the purpose of the experiment was to evaluate how well the hypernyms performed as labels, all dish names that were not in the final set of cluster candidates (without the non-cluster terms) were removed. Since our annotators only assigned a single label to each of the terms in our evaluation sets we were not able to directly compare the class labels of each term to the assigned label of its cluster. Instead, we used a list of seed terms related to the category "food". Since each cluster's label contains a list of K (5) words we select all clusters for which at least one of the K label words is in our seed set. We compare this list of food terms to the set of food terms in our evaluation set.

For our "food" seed set we chose (12) seeds relating to food (see Table 3.6 for the complete list of seeds). In order to calculate precision from test set 2 we filtered all terms that did not appear at least once in any of the 600 reviews. We, then, calculated precision and recall as follows:

Let D be the set of dishes extracted from the annotations and let S be the set of dishes returned by the cluster labeling method

$$Precision = \frac{\|D \cup S\|}{\|S\|} \tag{4.5}$$

$$Recall = \frac{\|D \cup S\|}{\|D\|} \tag{4.6}$$

The following table summarizes precision and recall measurements for the evaluation sets

4.3 Aspect Based Sentiment

In Section 3.4 we proposed aspect based sentiment analysis as an application of our proposed method. Due to the large number of reviews discussing food, we chose to evaluate the "food" aspect. Our dataset for evaluation consisted of the set of 600 annotated reviews randomly selected from the larger review corpus. We found that there were a number of ambiguous phrases in the annotations where our annotators could have chosen to split a phrase into two separate entities. In order to account for this we evaluated both partial and non-partial matches. Our algorithm for matching is described in Figure 4.1

4.3.1 Kulback Leibler Divergence

Kulback Leibler divergence (KLD) is defined as the relative entropy between two probability distributions (Losee, 1999; Carpineto, 2001; Vechtomova, 2010). We compute KLD using the following equation:

$$KLD(X) = P(X) \log \frac{P(X)}{(Q(X)} \tag{4.7}$$

Here, P represents the probability that a term X appears in the relevant set and Q represents the probability that X appears in the non-relevant set. In order to use KLD as a measure of polarity we use the set of words that appear in reviews containing a 10 star rating as the positive (relevant) set and all the words that appear in reviews containing less than a 3 star rating as the negative (relevant) set. The polarity of each term is

$$Polarity(x) = \begin{cases} Positive & \text{if KLD(x)} > 0 \\ Negative & \text{if KLD(x)} < 0 \end{cases} \tag{4.8}$$

For each method we computed precision and recall for each review and the average precision and recall over all the reviews. Tables 4.6 and 4.7 summarizes the results of our experiment. For our baseline we chose the second method proposed by (Verma and Bhattacharyya, 2009), described in Section 3.4.4.

Figure 4.1: Pseudocode for evaluating aspect oriented sentiment

```
1: for all reviews in the annotated set do
2:     Identify all exact matches between extracted food (from clusters) and annotated set
3:     for all extracted terms exactly matching an annotated term do
4:         Identify exact matches between the annotated modifier and the extracted modifier.
5:         for all extracted modifiers that matches an annotated modifier exactly do
6:             if Polarity(extracted modifier) = Polarity(annotated modifier) then add tuple (modifier, food) to the relevant set
7:             end if
8:         end for
9:         for all partial matches on modifiers where the annotated modifier subsumes the extracted modifier do repeat 6
10:        end for
11:        for all partial matches on modifiers where the extracted modifier subsumes the extracted modifier do repeat 6
12:        end for
13:    end for
14:    for all partial matches on food where the extracted food subsumes the annotated food do repeat 2 to 12
15:    end for
16:    for all partial matches on food where the annotated food subsumes the extracted food do repeat 2 to 12
17:    end for
18: end for
```

Table 4.6: Comparison of Precision for various Polarity Score Calculations

Method	Average Precision	Precision P-Value
Max over Synsets	0.3452	N/A
Weighted Average over Synsets	0.3569	0.4198
Average over Synsets	0.3690	0.07909
KLD	0.3870	0.005

Table 4.7: Comparison of Recall for various Polarity Score Calculations

Method	Average Recall	Recall P-Value
Max over Synsets	0.2338	N/A
Weighted Average over Synsets	0.3087	2.07E-10
Average over Synsets	0.3173	7.27E-14
KLD	0.3998	< 2.2E16

Although KLD had the best performance out of the four methods, its major drawback is that due to its binary nature there is no simple method to handle the case of objective modifiers. One possible suggestion is to treat those terms that had KLD scores around 0 to be objective with the assumption that objective modifiers are just as likely to occur in the positive set as the negative set. However, due to the small number of negative reviews many terms appear positive simply due to the fact that they don't appear in the negative set. Another possible solution to this would be to have an objective set to compute KLD against. This is difficult to obtain since most situations where food is discussed; modifiers are used in a subjective manner; therefore we leave it to future work to improve on this.

4.4 Comparison to Multigrain Topic Modeling

We compared our clustering technique to the method based on multi-grain topic modelling proposed in Titov and Macdonald (2008) (see Section 2.2.3.4.3) on our corpus of 157,865 reviews. In order to compare the two methods, we performed a similar set of preprocessing steps to those in (Titov and Macdonald,2008). Titov and Macdonald (2008) state that their method requires that the number of global topics is at least double the number of local topics. We chose to use a configuration similar to their restaurant configuration (19 local, 50 global). They claim that the quality of the local topics is not affected by the number of global topics as long as the number of global topics meets the requirement stated above, however, they do not provide any reasoning for the choice for the number of local topics. For our evaluation we used an open source implementation of the methods proposed in Titov and Macdonald, written by Masanao Ochi [1].

[1]https://github.com/m-ochi/mglda/blob/master/

Table 4.8: Cluster F_1 scores for MGLDA vs Hierarchical Clustering

Method	F_1 score
Hierarchial Clustering (0.02 cut-off)	0.330
Hypernym Clustering (0.02 cut-off)	0.429
MGLDA Soft Clustering (304 cut-off)	0.378
MGLDA Hard Clustering (97 cut-off)	0.264

In order to compare our methods against MGLDA, we had to address to problems. The first problem was that MGLDA results in a distribution over the terms for each topic. When comparing the soft clustering against our hard clustering, the soft clustering had the advantage that terms could be counted towards the precision and recall in multiple clusters. In order to address this, we also compared against a hard clustering version. The hard clustering was computed by assigning a term to the topic that had the highest probability of generating that term. For example if there were two topics T_1 and T_2 and $P(restaurant|T_1) = 0.5$ and $P(restaurant|T_2) = 0.3$ then restaurant would be assigned to cluster T_1. This can be written , more formally , using the following equation:

Let T be the set of Topics and let w be a word

$$Cluster(w) = \underset{t \in T}{\operatorname{argmax}} P(w|t) \tag{4.9}$$

The second problem is that each word in the corpus is included is given a topic probability. That is each word is included in the clustering. Since our evaluation set only contained cluster candidates our method had an advantage in terms of overall cluster precision. In order to address this as we only considered the top k words in each topic distribution as a cluster and we only considered cluster candidate terms. All other terms were ignored when computing cluster f measure. Table 4.8 summarizes the results our our comparison.

Table 4.8 shows that after hypernym clustering our method has a higher cluster F_1 score vs both the soft clustering and hard clustering versions of MGLDA. On the other hand,the soft clustering version of MGLDA did outperform clustering based on Lin's similarity. It should be noted that Titov and Macdonald (2008) also found that MG-LDA did not perform as well on the restaurant reviews. They suggested that this was likely caused by the fact that restaurant reviews are small in size, only 3 or 4 sentences on average. Furthermore the reviews in our corpus were made up of a variety of restaurant types such as Chinese, Japanese and Italian. Titov and Macdonald (2008) suggest that the performance of MGLDA might be better on restaurant reviews belonging to a specific type of restaurant.

Chapter 5

Conclusion and Future Work

In this thesis we proposed an unsupervised method for discovering semantic classes related to aspects in consumer reviews. Our method was based on the underlying assumption that terms with similar meanings appear in similar contexts (Lin, 1998). Our methods do not require the creation of any training sets, nor do they require prior knowledge of the semantic classes present in the corpus. We introduced a two-step hierarchical clustering process based on semantic similarity and hypernym similarity and demonstrated that it greatly increases the cluster F_1 score over a single step clustering process. We also proposed a solution for automatically assigning class labels to our clusters, one of the major shortfalls of unsupervised methods. Finally, we demonstrated how our method for semantic class discovery could be used as part of an application for determining the sentiment of rateable aspects contained in restaurant reviews.

Our experiments have provided a number of key insights into the problem of aspect discovery. The first is that the assumptions proposed by Lin (1998) that terms that appear in similar contexts have similar meanings holds true. We showed that this is especially true if the contexts are based on the types of terms used to modify rateable aspects. In other words, consumers use different terms to modify different types of aspects and these terms can be used to identify the different aspects. The work in this thesis focussed on a small set of modifier dependency triples and due to this did not cover use cases where a verb carries the polarity, for example in the phrase “the restaurant rocked” the relationship between “rocked” and “restaurant” would not have been taken into account by our method. This is unlikely to affect our system given a large corpus because if the term ”restaurant“ was important word representing a particular review aspect, it would have likely appeared in a relationship with one of our modifier dependencies, however, this may be problematic given a small corpus. Future experiments will need evaluate if our methods are still effective on smaller corpora and how much of an effect these non-adjective modifiers have.

The second insight is that hierarchical clustering is an effective clustering approach for aspect discovery due to the hierarchical nature of aspects. During the clustering process we found that intermediate clusters formed sub aspects. An example of this would be in the food category where clusters involving seafood, meats, desserts etc. were created before coming together into larger clusters. The hierarchical nature of aspects is also highlighted by the ability of cluster hypernyms to be used as class labels. In this case the hypernym labels acted as the highest level in the term hierarchy. Future experiments should compare hierarchical clustering to other non-hierarchial clustering methods such as K-Means in order to affirm that presence of an aspect class hierarchy does give hierarchical clustering an advantage over non hierarchical methods

Our methods do have a number of shortcomings that must be addressed in future work. First, we extracted all nouns as initial candidates which resulted in a large number of candidates. Many of these candidates had spelling errors which led to a large number of “junk” clusters. The reason being is that spelling errors in both the candidates and the modifiers would have looked like completely different relationships to our system. We did perform some preprocessing in order to address this, but we did not focus on this as part of our system. Any live application would have to deal with spelling errors so it is important that future works are able to address their presence and their effect on the accuracy of their system. The second shortcoming of our method is that it resulted in a large number of small fine grained classes even after clustering the hypernym representations of the clusters. Although, human reviewing of the final clusters is manageable, it still could be a cumbersome task for the developer of an application. One possible way to reduce this would be to recursively continue the process of clustering based on hypernyms. We found that a number of the clusters had hypernyms that were hyponyms of the hypernyms in some of the other clusters, for example the hypernym “salad” is actually a hyponym of “dish” or “food”. Another potential shortcoming is that our method for hypernym gathering involves the use of a context word for resolving ambiguity. While we believe that this does not change the supervision level of our method, a system for automatically determining this based on the terms that appear in the reviews would be ideal.

Currently we have only evaluated our methods on restaurant reviews. We would also like to evaluate our methods on other common review corpora such as hotels, and products. Since we do not make any assumptions specific to our domain, except for those that are universal to all consumer reviews, we believe that our methods should scale. Furthermore, previous works such as Titov and Macdonald (2008) have shown to have better results on hotel and product reviews as compared to restaurant reviews.

References

[1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 85–94, New York, NY, USA, 2000. ACM.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. of 20th International Conference on VLDB*, pages 487–499, 1994.

[3] Mohamad H Ahmadi. *A semi supervised approach to the construction of semantic lexicons*. PhD thesis, University of Waterloo, 2012.

[4] David Andrzejewski and Xiaojin Zhu. Latent dirichlet allocation with topic-in-set knowledge. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, SemiSupLearn '09, pages 43–48, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[5] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data mining*, KDD '02, pages 436–442, New York, NY, USA, 2002. ACM.

[6] Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 57–64. Association for Computational Linguistics, 1999.

[7] Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *In Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, 1997.

[8] Sasha Blair-goldensohn, Tyler Neylon, Kerry Hannan, George A. Reis, Ryan Mcdonald, and Jeff Reynar. Building a sentiment summarizer for local service reviews. In *In NLP in the Information Explosion Era*, 2008.

[9] David Blei and John Lafferty. Topic models. *Text Mining: Theory and Applications*, 2009.

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[11] Andrew Eliot Borthwick. *A maximum entropy approach to named entity recognition.* PhD thesis, New York, NY, USA, 1999. AAI9945252.

[12] Gerlof Bouma. Normalized mutual information in collocation extraction. 2009.

[13] Sergey Brin. Extracting patterns and relations from the world wide web. In *Selected Papers from the International Workshop on The World Wide Web and Databases*, WebDB '98, pages 172–183, London, UK, UK, 1999. Springer-Verlag.

[14] Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 804–812, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[15] Sharon A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 120–126, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[16] Claudio Carpineto, Renato de Mori, Giovanni Romano, and Brigitte Bigi. An information-theoretic approach to automatic query expansion. *ACM Trans. Inf. Syst.*, 19(1):1–27, January 2001.

[17] Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. Equations for part-of-speech tagging. In *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789, 1993.

[18] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Comput. Linguist.*, 16(1):22–29, March 1990.

[19] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *In proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 449–454, 2006.

[20] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure trees. In *LREC*, 2006.

[21] Weifu Du and Songbo Tan. An iterative reinforcement approach for fine-grained opinion mining. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 486–493, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[22] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, 19(1):61–74, March 1993.

[23] Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06*, pages 417–422, 2006.

[24] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *Proceedings of the 13th*

International Conference on World Wide Web, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.

[25] Guohong Fu and Kang-Kwong Luke. Chinese named entity recognition using lexicalized hmms. *SIGKDD Explor. Newsl.*, 7(1):19–25, June 2005.

[26] Rayid Ghani and Rosie Jones. A comparison of efficacy and assumptions of bootstrapping algorithms for training information extraction systems. In *Workshop on Linguistic Knowledge Acquisition and Representation at the Third International Conference on Language Resources and Evaluation (LREC*, 2002.

[27] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Comput. Linguist.*, 32(1):83–135, March 2006.

[28] Thomas . L. Griffiths and Mark. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(Suppl. 1):5228–5235, April 2004.

[29] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[30] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[31] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM.

[32] Sean P. Igo and Ellen Riloff. Corpus-based semantic lexicon induction with web-based corroboration. In *Proceedings of the Workshop on Unsupervised and Minimally Supervised Learning of Lexical Semantics*, UMSLLS '09, pages 18–26, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[33] Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1035–1045, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[34] Wei Jin and Hung Hay Ho. A novel lexicalized hmm-based learning framework for web opinion mining. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 465–472, New York, NY, USA, 2009. ACM.

[35] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[36] Julian Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech and Language*, 6(3):225 – 242, 1992.

[37] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[38] Sang-Zoo Lee, Jun-ichi Tsujii, and Hae-Chang Rim. Lexicalized hidden markov models for part-of-speech tagging. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, COLING '00, pages 481–487, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[39] Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. Structure-aware review mining and summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 653–661, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[40] Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 768–774, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[41] Dekang Lin. Dependency-based evaluation of minipar. *Workshop on the Evaluation of Parsing systems*, pages 317–330, 1998.

[42] Dekang Lin and Patrick Pantel. Induction of semantic classes from natural language text. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 317–322, New York, NY, USA, 2001. ACM.

[43] Jianhua Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, 1991.

[44] R. M. Losee. *The science of information: Measurements and applications.* Academic Press Prof., Inc, 1990.

[45] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[46] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[47] Arjun Mukherjee and Bing Liu. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 339–348, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[48] Business Development Bank of Canada. Mapping your future growth five game-changing consumer trends, 2013.

[49] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, KDD '02, pages 613–619, New York, NY, USA, 2002. ACM.

[50] Patrick Pantel and Deepak Ravichandra. Automatically labeling semantic classes. In *Proceedings of HLT/NAACL 2004*, 2004.

[51] Marius Pasca. Acquisition of categorized named entities for web search. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 137–145, New York, NY, USA, 2004. ACM.

[52] Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 339–346, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[53] Katharina Probst, Rayid Ghani, Marko Krema, Andrew Fano, and Yan Liu. Semi-supervised learning of attribute-value pairs from product descriptions. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2838–2843, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[54] Vasin Punyakanok and Dan Roth. The use of classifiers in sequential inference. In *NIPS*, pages 995–1001. MIT Press, 2001.

[55] Luole Qi and Li Chen. A linear-chain crf-based learning approach for web opinion mining. In Lei Chen, Peter Triantafillou, and Torsten Suel, editors, *Web Information Systems Engineering – WISE 2010*, volume 6488 of *Lecture Notes in Computer Science*, pages 128–141. Springer Berlin Heidelberg, 2010.

[56] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1199–1204, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[57] Lawrence R. Rabiner. Readings in speech recognition. chapter A Tutorial on hidden markov models and selected applications in speech recognition, pages 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[58] Santosh Raju, Prasad Pingali, and Vasudeva Varma. An unsupervised approach to product attribute extraction. In *Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, ECIR '09, pages 796–800, Berlin, Heidelberg, 2009. Springer-Verlag.

[59] Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 41–47, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[60] Ellen Riloff and Jessica Shepherd. A corpus-based approach for building semantic lexicons. In *In Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, pages 117–124, 1997.

[61] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[62] Brian Roark and Eugene Charniak. Noun-phrase co-occurrence statistics for semiautomatic semantic lexicon construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 1110–1116, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics.

[63] Christopher Scaffidi, Kevin Bierhoff, Eric Chang, Mikhael Felker, Herman Ng, and Chun Jin. Red opal: Product-feature scoring from reviews. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, EC '07, pages 182–191, New York, NY, USA, 2007. ACM.

[64] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 134–141, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[65] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, Cambridge, MA, 2005.

[66] Benjamin Snyder and Regina Barzilay. Multiple aspect ranking using the good grief algorithm. In *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL*, pages 300–307, 2007.

[67] Steffen Staab. Learning concept hierarchies from text with a guided agglomerative clustering algorithm. In *Proceedings of the Workshop on Learning and Extending Lexical Ontologies with Machine Learning Methods*, 2005.

[68] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[69] Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction, 2004.

[70] Charles Sutton and Andrew McCallum. *Introduction to Conditional Random Fields for Relational Learning.* MIT Press, 2006.

[71] Michael Thelen and Ellen Riloff. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 214–221, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[72] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057v1*, 1999.

[73] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 111–120, New York, NY, USA, 2008. ACM.

[74] Takashi Tomokiyo and Matthew Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment - Volume 18*, MWE '03, pages 33–40, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[75] Peter D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 491–502, London, UK, UK, 2001. Springer-Verlag.

[76] Olga Vechtomova. Facet-based opinion retrieval from blogs. *Inf. Process. Manage.*, 46(1):71–88, 2010.

[77] Olga Vechtomova. A method for automatic extraction of multiword units representing business aspects from user reviews. *Journal of the American Society for Information Science and Technology. In press*, 2013.

[78] Olga Vechtomova and Stephen E. Robertson. A domain-independent approach to finding related entities. *Inf. Process. Manage.*, 48(4):654–670, 2012.

[79] Shitanshu Verma and Pushpak Bhattacharyya. Incorporating semantic knowle dge for sentiment analysis. In *In Proceedings of the 6th International Conference on Natural Language Processing (ICON)*, 2009.

[80] Martin Wainwright, Tommi Jaakkola, and Alan Willsky. Tree-based reparameterization framework for approximate estimation of stochastic processes on graphs with cycles, 2001.

[81] John S Whissel. *Evaluating Clusterings by Estimating Clarity.* PhD thesis, Univeristy of Waterloo, 2012.

[82] Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210, 2005.

[83] Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in*

Natural Language Processing: Volume 3 - Volume 3, EMNLP '09, pages 1533–1541, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[84] Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. Aspect ranking: identifying important product aspects from online consumer reviews. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1496–1505, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[85] Zhongwu Zhai, Bing Liu, Hua Xu, and Peifa Jia. Constrained lda for grouping product features in opinion mining. In *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, PAKDD'11, pages 448–459, Berlin, Heidelberg, 2011. Springer-Verlag.

[86] Lei Zhang, Bing Liu, Suk Hwan Lim, and Eamonn O'Brien-Strain. Extracting and ranking product features in opinion documents. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1462–1470, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.